Recall: For Loops

- **loop sequence:** grades
- **loop variable:** x
- **body:** print(x)

To execute the for-loop:
1. Check if there is a “next” element of loop sequence
2. If so:
   - assign next sequence element to loop variable
   - Execute all of the body
   - Go back to Line 1
3. If not, terminate execution.

### Different types of Repetition

1. **Process each item in a sequence**
   - Compute statistics for a dataset.
   - Send all your contacts an email.
2. **Do something n times**
   - Draw a checkers board.
   - Run a protein-folding simulation for $10^6$ time steps.
3. **Do something an unknown number of times**
   - Fly up until you’re near the ceiling.
   - Play hangman until 6 strikes.

### Beyond Sequences: The while-loop

- **Relationship to for-loop**
  - Broader notion of “keep working until done”
  - Must explicitly ensure condition becomes false
  - You explicitly manage what changes per iteration

### While-Loops and Flow

```python
import random
num = random.randint(0,10)
guessed_it = False
print('I’m thinking of a number.')
guessed_it = False
while not guessed_it:
    guess = input('Guess it: ')  
guessed_it = (num == guess)  
    print('Well done!')
```

### Q1: What gets printed?

<table>
<thead>
<tr>
<th>a = 0</th>
<th>a = 0</th>
<th>a = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>while a &lt; 1:</td>
<td>while a &lt; 2:</td>
<td>while a &gt; 2:</td>
</tr>
<tr>
<td>a = a + 1</td>
<td>a = a + 1</td>
<td>a = a + 1</td>
</tr>
</tbody>
</table>

```python
print(a) | print(a) | print(a) |
```

4/11/18
Q2: What gets printed?

```python
a = 4
while a > 0:
    a = a - 1
print(a)
```

A: Infinite Loop!
B: 4
C: 4
D: I don’t know

for vs. while

do something n times

```python
for k in list(range(n)):
    # do something

k = 1
while k < n:
    # do something
    k = k + 1
```

```
My preference? for-loop
```

Remember Hangman?

```python
import random, hangman
word_list = [... words we want user to guess ...]
N_GUESSES = 10
secret = hangman.SecretWord(random.choice(word_list))

for n in list(range(N_GUESSES)):
    secret.word_so_far()
    user_guess = input("Guess a letter: ")
    secret.apply_guess(user_guess):
    if secret.is_solved():
        print("YOU WIN!!!")
        break
    #jumps out of the for-loop
secret.reveal()```

Using while-loops Instead of for-loops

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Better for modifying data</td>
<td>Infinite loops more likely</td>
</tr>
<tr>
<td>More natural than range</td>
<td>Easy to forget loop vars</td>
</tr>
<tr>
<td>Works better with deletion</td>
<td>Or get stop condition wrong</td>
</tr>
<tr>
<td>Better for convergent tasks</td>
<td>Require more management</td>
</tr>
<tr>
<td>Loop until calculation done</td>
<td>Initialize the condition?</td>
</tr>
<tr>
<td>Exact steps are unknown</td>
<td>Update the condition?</td>
</tr>
<tr>
<td>Easier to stop early</td>
<td></td>
</tr>
<tr>
<td>Just set loop var to False</td>
<td></td>
</tr>
</tbody>
</table>